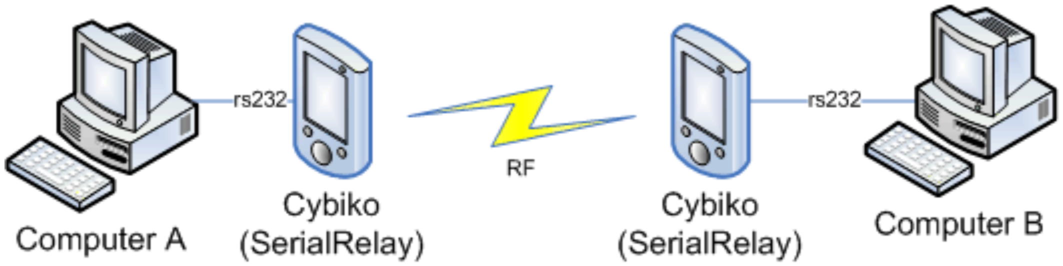
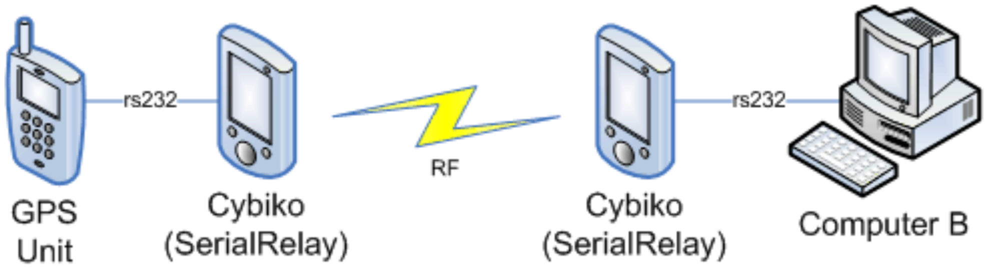
Serial Relay

The serial relay project allows the serial ports on a Cybiko Classic to be virtually linked with another, in full-duplex mode.



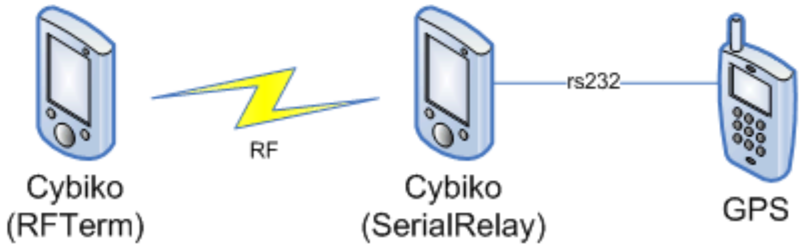
The program runs in a point-to-point mode, that is only communication between two Cybiko's is allowed. However, the range of the serial relay solution can be extended by the use of the Repeater program.

It doesn't always have to be a computer on the other end consider this interesting idea.



This would allow a computer to remotely monitor a GPS unit that is moving around sending back it's coordinates.

When used with RFTerm the remote VT100 terminal emulator you can get serial access to any device.



This would show all the NMEA strings that are being transmitted by the GPS unit onto another cybiko

This program will only work on the Classic as it requires an RS232 port. You will also need the comport.dl driver.

Usage

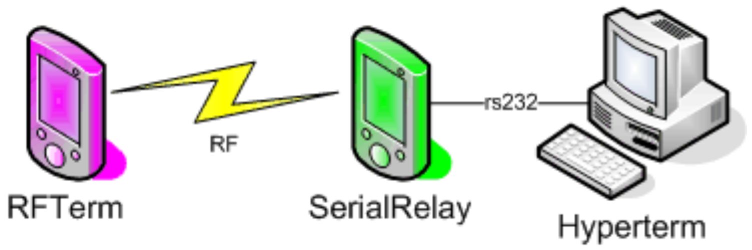
From the main screen several keyboard options are available, this help is also available by pressing the **?** key on the cybiko.

Key	Description
Select	locate other Cybiko
Enter	For options screen - Options are described below
I	Information
L	Toggle line numbers when in debug mode
C	Clear console
D	Toggle debugging mode
Esc	to exit.

This program has been tested with kermit (http://www.columbia.edu/kermit/) transferring a 1.3Mb file and it worked perfectly. It should work for other serial protocol; xmodem, ymodem, zmodem, ppp, slip etc..

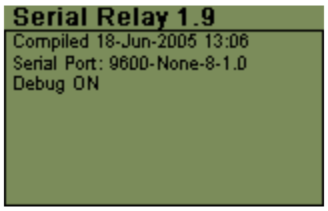
SerialRelay to RFTerm Walkthrough

In this demonstration we will be connecting two cybiko to each other using RFTerm and SerialRelay. The cybiko have been given the nicknames: purple and green



Step 1

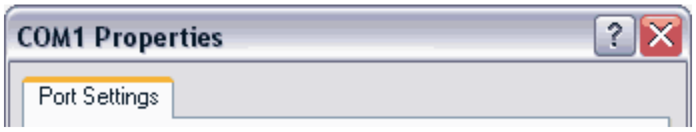
Starting up SerialRelay on green. When you first run the program you will see a screen similiar to that below. The **D** key has been pressed to enable debug tracing.

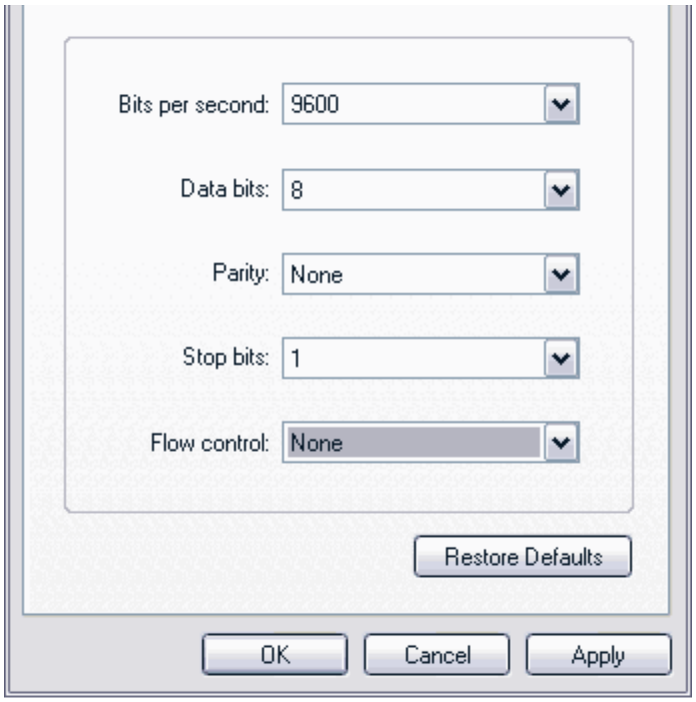


```
Serial Relay 1.9
Compiled 18-Jun-2005 13:06
Serial Port: 9600-None-8-1.0
Debug ON
```

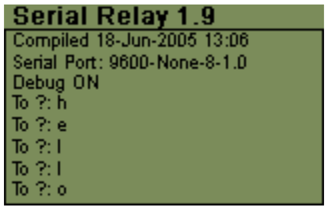
Step 2

Start hyperterminal or some other serial communication program on the PC, make sure you select the correct port that you have wired up the green cybiko to. The following hyperterminal settings match those of the default serial relay startup. That being 9600-n-8-1.



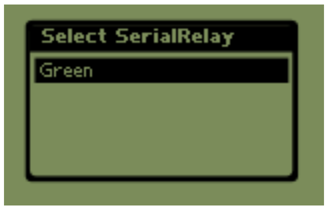

Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

To verify that you at least have connectivity and all the serial port parameters are set correctly. We can type the string **hello** into hyperterminal this should appear on the serial relay console, it won't be sent to any remote cybiko at this point in time as we have not selected any. So **To** will appear as a **?** question mark.

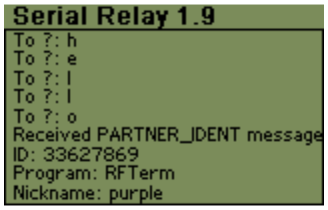

```
Serial Relay 1.9
Compiled 18-Jun-2005 13:06
Serial Port: 9600-None-8-1.0
Debug ON
To ?: h
To ?: e
To ?: l
To ?: l
To ?: o
```

Step 3

Now to get some connectivity happening, on a 2nd cybiko (purple) fire up the RFTerm program and press the **SELECT** button.

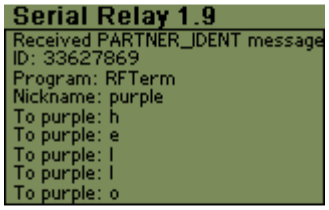


This will present you with a dialog of all other cybiko that are running. Locate the name of the one running the SerialRelay program and press **ENTER**. Once you have done this the cybiko running serialRelay (green) will show some information about this connection.

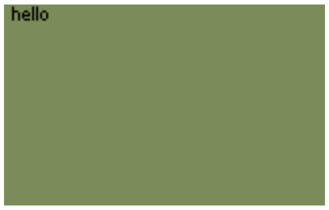

```
Serial Relay 1.9
To ?: h
To ?: e
To ?: l
To ?: l
To ?: o
Received PARTNER_IDENT message
ID: 33627869
Program: RFTerm
Nickname: purple
```

Step 4

Proving the connection works. Now return to the hyperterminal session once more we type the string **hello** this time as serialrelay now has a connection with RFTerm the characters will be transported from the cybiko connected to the PC onto the cybiko running RFTerm.

SerialRelay(Green) → 

```
Serial Relay 1.9
Received PARTNER_IDENT message
ID: 33627869
Program: RFTerm
Nickname: purple
To purple: h
To purple: e
To purple: l
To purple: l
To purple: o
```

RFTerm(Purple) → 

```
hello
```

Now that it is all working press **D** to turn off debug mode. There is a slight overhead when running in debug mode that will throttle your effective baud rate.

Options

- **Partner Type** - Either serialrelay or repeater. Note when you select repeater - autoJoin will be automatically set to ON, as it makes no sense for it to now be OFF.
- **Baud Rate** - Communication speed with host: 110, 300, 600, 1200, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200. Defaults to 9600.
- **Data Bits** - Size of word size: 5,6,7,8 - Defaults to 8.
- **Stop Bits** - 1,1.5,2 - Defaults to 1.
- **Parity** - even, odd, none - Default to NONE.
- **Handshake** - none, CTS, XON/XOFF - Default to NONE.
- **Power Mode** - Invokes an RF power hack to increase a units range.
- **Duplex Mode** - Full, Tx Only, Rx Only.
- **AutoJoin** - Autojoin affect the process of the <SELECT> key. If the autojoin flag is set to YES/ON then the program on the other end must already be running for it to be notified. If it's not the CyOS wont automatically start it for you when it gets a message.
  When AutoJoin=OFF and the application on the other end is not running then you are prompted to start it by the CyOS. This mode only makes sense when serialrelay is talking to itself.
- **Reboot Mins** - The unit will automatically reboot itself after this many minutes have elapsed. This can be used in concert with Desktop replacement with an auto runnable program.
- **Reboot Send** - After a given number of bytes are transmitted via RF the unit will auto reboot.
- **Reboot Rcvd** - After a number of bytes have been received via RF reboot.
- **Poll Freq(ms)** - This option Allows the frequency as which the serial port is polled to be adjusted. What is does is introduce a wait time into the time that the message loops wait for an RF message.
- **RF silence** - that amount of time in minutes that has to elapse after RF activity before the unit can be considered ready for reboot. This setting runs in parallel with the reboot Mins setting. For example if the unit is told to reboot after 10 mins, and it requires 2min of RF silence. If a packet is received at 9mins then the unit will wait until 11min before rebooting.
- **Partner quit notify** (v1.10) - Should a dialog be displayed when the partner cybiko quits. on/off - Default on.

Comments

1. I'm assuming that the scrolling output on the screen has no '*history*' and that you can't ask the cybiko to scroll back thru historic '*off screen*' content? - This would obviously eat memory like mad.
2. If these units are to be deployed in '*highish availability*' apps, we will need to take an approach rather like that of the apache webserver, where a thread would kill itself after performing a predefined number of operations.
3. Idea. We have a tuneable parameter, '*maxops*'. Every time we do a Tx or Rx (or screen operation), maxops is decremented. If it hits zero, the Cybiko issues an XOFF. It then automatically does a hard reset on itself. On reboot, (assuming we've cracked the '*auto-run this app*' piece), the app notices that it has just come back from a self-initiated reset & issues an XON. Of course, whatever is attached to the Cybiko serial port will have to be designed and coded to work with this behaviour.
4. Thought. How about a function call that prints the available free RAM of the cybiko to the screen, this would help when trying to find out where leaks are coming from.
5. Also, a client initiated '*just to a reboot now, please*' command. This would allow a smart client to decide when it was a suitable time to have the Cybiko reboot.
6. A hardware based solution. We can find out how to cause the little LED on the cybiko to flash from red to yellow (wireless traffic does this anyway). Our app would flash this LED every so often, like it was servicing a watchdog timer. Have a little RC circuit that notices if the LED stops flashing. If it sees this, it just pushes the '*reset*' button.
7. Point of interest. Whilst running serialrelay, I powered up a 3rd Cybiko. There was much buzzing and beeping when the serialrelay units noticed the presence of the new cybiko. Data xmission was unaffected - which is good. Interesting that this behaviour ran '*in the background*'. Is this hardwired into the OS (Operating System), or can we do this too?
8. DK to introduce hard logging (with time/datestamp) onto the PC test harness programs so as to determing when things actually stopped. Must also try to characterise what brings on hangups - messagelength, repeat rates, baud rate etc, etc
9. The fact that these ran with continual 2-way serial data traffic for (at least) over 6 hours is actually very encouraging.
10. Point from Brett - Perhaps explore the use of the vibrator function to act as a way to drive the reset line on the Cybiko.
11. If this project is still alive, what about writing a 'lite' version which has no options or interface, but purely forwards the RS232 port data?

Code

Code
serialrelay.zip v1.11
serialrelay-v10.zip v1.10
serialrelay_1.9.zip

Feature Request

- The enhanced debug more that displays control chars and allows line numbers is great. How about another debug option to display the time at which the message was sent/received?
- Auto restart. As we know, there are leaks in the CyOS serial port management code. An option to force a conmplete reset of the unit after a given uptime was exceeded would be very handy for 'embedded apps'. Clearly, this would have to be used in conjunction with an auto-launch of the serial relay application. Some sort of 'auto-rejoin' feature might also be needed.

Done - Version 1.8

- An option to cause an XOFF to be sent just before a unit does a reboot, and another option to cause an XON to be sent when a unit is recovering from a reset.

Change Log

22-Oct 1.11

- Publish / Subscriber implementation - this means that multiple clients can connect to a single serial relay cybiko. Data received by the master (serial relay) cybiko will not be forward to others, but anything received by the serial relay unit via it's serial port will be sent to everybody.
- Connecting partners now display CYID as printed on the back of the units.

4-Oct 1.10

- Partner quit dialog option setting - useful when you are running serial relay and the partner can come and go, without having to acknowlege the fact. Programs such as wmpremote and cameracontrol
- Partner quit did not clear the partner_info.id flag so serial relay would continue to try and send messages even thou it knows the partner has gone.

18-Jun 1.9

- Repeater connectivity added
- Alternative reboot vector for different model classics

14-Jun 1.8

- Auto reboot functionality with partner restablishment
- Adjustable polling frequency
- Streamlined option saving (internal only)
- 'I' key displays useful information.

6-Jun 1.7

- Configuration file is now held externally
- Added Full/Tx Only/Rx Only duplex setting. Note debug will still be displayed as character received in/out of cybiko but the transmission will not occur.
- Enhanced GUI (Graphical User Interface) & Feedback messages, expanded ctrl chars in debug mode

- Missing partner detection (don't loopback messages)

3-Apr 1.6

- Adjusting baud setting takes affect without needing app restart
- Additional message PARTNER_IDENT to allow other programs to use it as a gateway.

14-Mar 1.5

- Increased buffers from 80 to 256
- Complete re-write of option setting page using modular components.

13-Mar 1.4

- Increased buffers from 32 to 80 bytes to prevent buffer overruns when transmitting GPS data at 4800 baud.

Documentation

This is a Window CHM file generated automatically by http://www.doxygen.org/ (http://www.doxygen.org/) that describes the internals of the code.

cybiko/serialrelay.txt Last modified: 2009/11/27 17:54 (external edit)